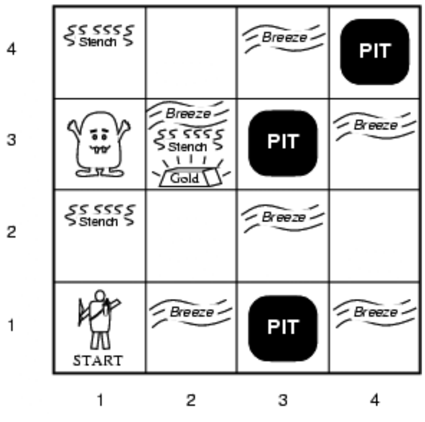
7 Logische Agenten

Die Wumpus-Welt



Sensoren:

Die Quadrate um den Wumpus herum stinken (Stench)

In Quadraten um eine Falltür (Pit) herum ist ein Luftzug (Breeze) zu spüren

Quadrat mit Gold glitzert (Glitter)

Läuft ein Agent gegen eine Wand, nimmt er einen Stoß wahr (Bump)

Wird das Wumpus getötet, dann stößt es einen Schrei aus (Scream)

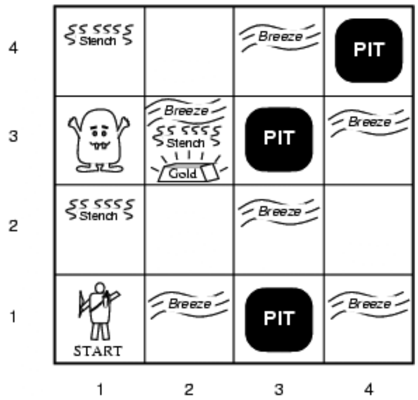
Wahrnehmungen in der Wumpus-Welt

Liste mit 5 Symbolen: [Stench, Breeze, Glitter, Bump, Scream]

Ausgangssituation:

[None, None, None, None, None]

Agent kann nach (2,1) oder (1,2) gehen, beide Felder sind sicher.



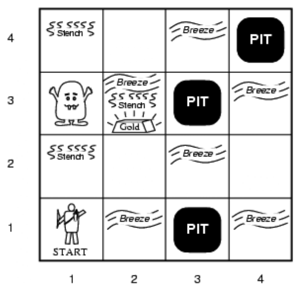
Agent startet in Feld (1,1)

Felder (2,1) und (1,2) sind sicher.

A. geht nach (2,1)

Wahrnehmung hier:

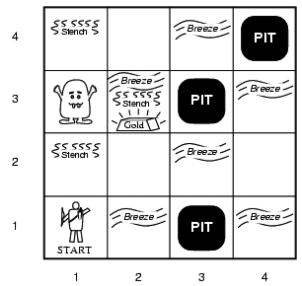
[None, Breeze, None, None, None]



Start
[None, None, None, None, None]

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 ok	2,2	3,2	4,2
1,1 A ok	2,1 ok	3,1	4,1

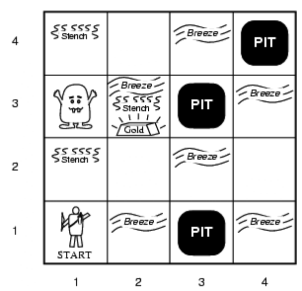
1. Zug: Agent geht nach (2,1)



1. Zug
[None, Breeze, None, None, None]

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 ok	2,2 P?	3,2	4,2
1,1 V ok	2,1 A B ok	3,1 P?	4,1

3. Zug: Agent geht nach (1,2)

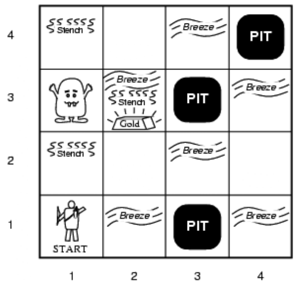


3. Zug
[Stench, None, None, None, None]

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A ok S	2,2 ok	3,2	4,2
1,1 V ok	2,1 V B ok	3,1 P!	4,1

5. Zug: Agent geht nach (2,3)

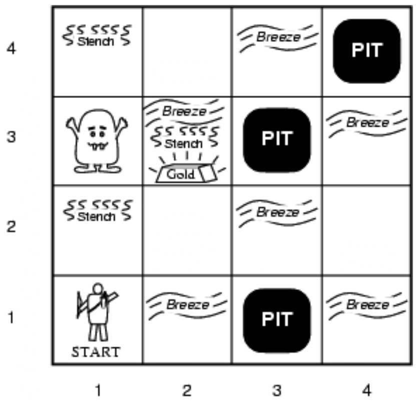
In jedem Fall, wo der Agent einen Schluss aus verfügbarer Information zieht, ist dieser Schluss garantiert korrekt, wenn die verfügbare Information korrekt ist.



5. Zug
[Stench, Breeze, Glitter, None, None]

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 $\boxed{A}$ S B G	3,3 P?	4,3
1,2 V S ok	2,2 V ok	3,2	4,2
1,1 V ok	2,1 V B ok	3,1 P!	4,1

Wahrheit in einer Welt



A_1: Das Wumpusfeld liegt neben einem Goldfeld

A_2: Jedes Luftzugfeld liegt neben einer Falle

Wahrheit

Die Sprache der Logik hat eine Semantik.

- Bsp.: Semantik arithmetischer Ausdrücke
  - $x+y = 4$ ist wahr, wenn $x = 2$ und $y=2$
  - falsch, wenn $x=1$ und $y=1$
- Die Semantik der Sprache der Logik definiert den Wahrheitsgehalt im Hinblick auf jede mögliche Welt (mögliche Welt = Modell).
- Sprechweise
- „m ist ein Modell von α" bedeutet „der Satz α ist im Modell m wahr"
- Modelle sind mathematische Abstraktionen, die festlegen, ob ein relevanter Satz wahr oder falsch ist.

Wahrheit und Logische Konsequenz

- Logische Konsequenz (entailment) zwischen Sätzen: ein Satz folgt logisch aus einem anderen
- Schreibweise: $\alpha \models \beta$
- $\alpha \models \beta$ gilt genau dann, wenn in jedem Modell, in dem α wahr ist, auch β wahr ist.
- Für die Wissensbasis KB erhalten wir
  - $KB \models \alpha$
  - α ist aus der Wissensbasis KB folgerbar (ableitbar)
- Formale Definition
  - $KB \models \alpha$, dann und nur dann, wenn in jedem Modell, in dem KB wahr ist auch α wahr ist

Beispiel in Wumpus Welt

Situation

1. Zug
[None, Breeze, None, None, None]

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 ok	2,2 P?	3,2	4,2
1,1 V ok	2,1 A B ok	3,1 P?	4,1

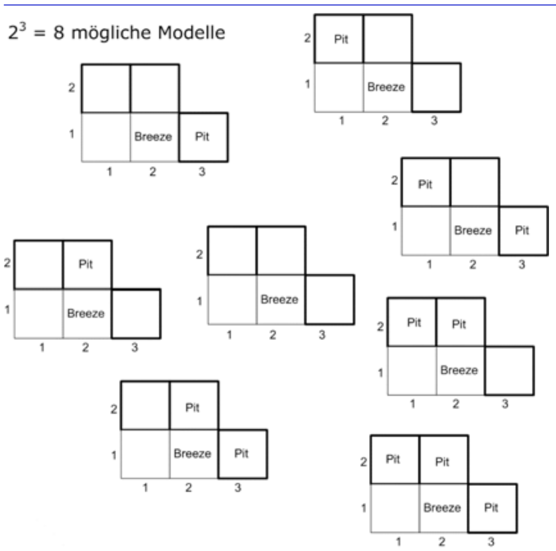
Agent hat in (1,1) nichts erkannt, aber in (2,1) einen Luftzug wahrgenommen.

Diese Wahrnehmungen, zusammen mit dem Wissen des Agenten über die Regeln der Wumpuswelt bilden die Wissensbasis (KB).

Der Agent will wissen, ob die benachbarten Quadrate (1,2), (2,2) und (3,2) eine Falltür enthalten; daher gibt es $2^3 = 8$ mögliche Modelle für das Vorhandensein von Falltüren.

Wahre u. falsche Modelle der Wissensbasis

$2^3 = 8$ mögliche Modelle



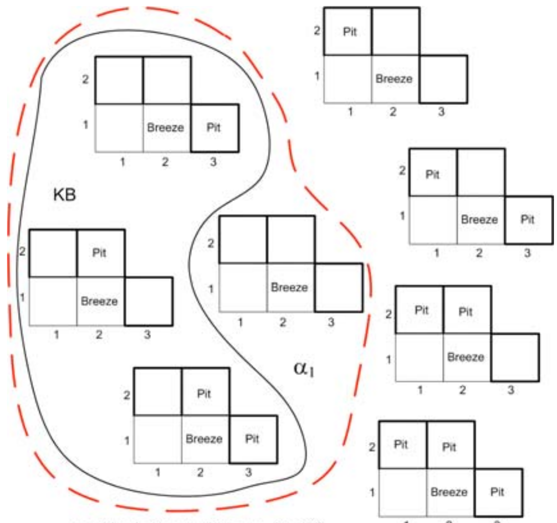
1. Zug
[None, Breeze, None, None, None]

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 ok	2,2 P?	3,2	4,2
1,1 V ok	2,1 A B ok	3,1 P?	4,1

Beobachtungen:

- nichts in (1,1)
- Luftzug in (2,1)

Wahre Modelle bei „keine Falltür in (1,2)“

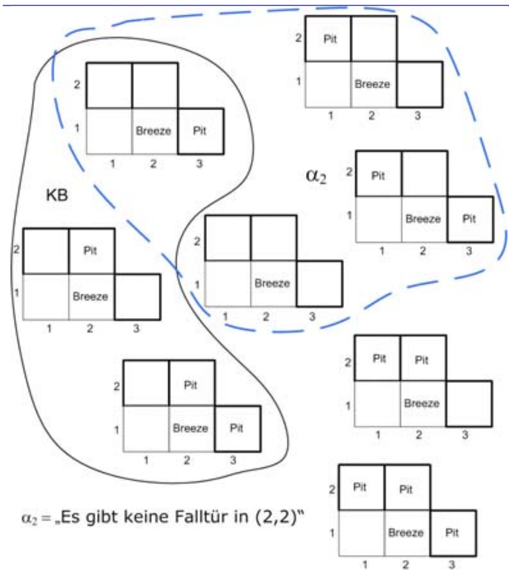


α_1 = „Es gibt keine Falltür in (1,2)“

Beobachtungen:

- nichts in (1,1)
- Luftzug in (2,1)

Wahre Modelle bei „keine Falltür in (2,2)“



α_2 = „Es gibt keine Falltür in (2,2)“

Beobachtungen:

- nichts in (1,1)
- Luftzug in (2,1)

Die Wissensbasis ist falsch in den Modellen, die dem widersprechen, was der Agent weiß, z. B. ist die KB falsch in jedem Modell, in dem (1,2) eine Falltür enthält, weil es keinen Luftzug in (1,1) gibt.

Es gibt 3 Modelle, in denen die Wissensbasis wahr ist.

Beobachtungen

1. In jedem Modell, in dem die Wissensbasis KB wahr ist, ist auch die Schlussfolgerung α_1 = „Es gibt keine Falltür in (1,2)“ wahr.

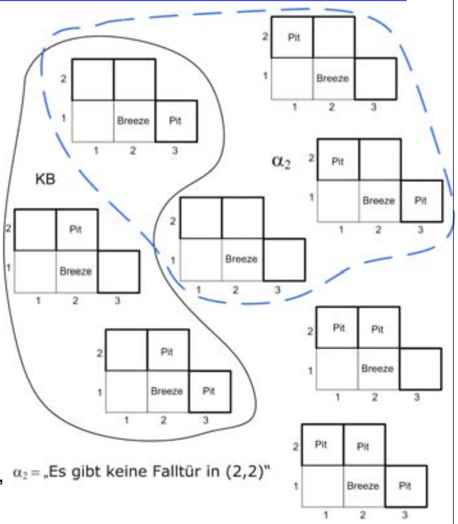
Damit ist α_1 eine logische Konsequenz der Wissensbasis KB

Schreibweise KB $|= \alpha_1$

2. In einigen Modellen, in denen die Wissensbasis wahr ist, ist die Schlussfolgerung α_2 falsch.

Damit ist α_2 keine logische Konsequenz von KB, Schreibweise KB $\neg|= \alpha_2$.

Das bedeutet: Der Agent kann nicht schließen, dass es in (2,2) keine Falltür gibt. (Er kann aber auch nicht schließen, dass es eine Falltür gibt.)



Logische Inferenzen

Durch die Anwendung der Definition einer logischen Konsequenz können Schlüsse gezogen werden, d.h. logische Inferenzen ausgeführt werden.

Ein Inferenz-Algorithmus listet alle Modelle auf, und prüft, ob eine Schluss α in allen Modellen der Wissensbasis wahr ist.

Wenn ein Inferenz-Algorithmus i den Schluss α aus der Wissensbasis KB ableiten kann, dann schreiben wir:

$$\mathbf{KB} \;|=_i \alpha$$

Sprechweisen:

α wird durch i von KB abgeleitet

i leitet α von KB ab

Eigenschaften von Inferenz-Algorithmen

- Korrekt (= wahrheitserhaltend)
- vollständig

  jeder folgerbare Satz kann abgeleitet werden

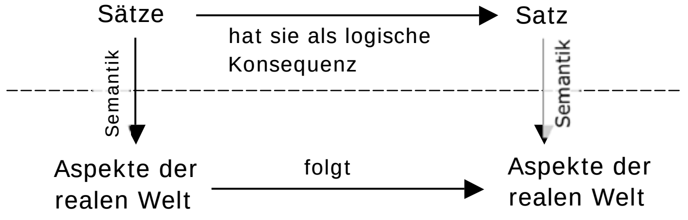
Wenn KB in der realen Welt wahr ist, dann ist auch jeder durch einen wahrheitserhaltenden Inferenz-Algorithmus von der KB abgeleitete Satz α in der realen Welt wahr.

Reale Welt und ihre Repräsentation

Inferenz-Prozess arbeitet mit der „Syntax“.

Dem *entspricht* der Prozess der Beziehungen in der realen Welt, dass ein Aspekt der realen Welt aufgrund anderer Aspekte der realen Welt der Fall ist (d.h. zutrifft).

Wittgenstein (1920) in *Tractatus logico-philosophicus*: „Die Welt ist alles, was der Fall ist.“



Syntax der Aussagenlogik

If S is a sentence, ¬S is a sentence (negation)

If S1 and S2 are sentences, S1 ∧ S2 is a sentence (conjunction)

If S1 and S2 are sentences, S1 ∨ S2 is a sentence (disjunction)

If S1 and S2 are sentences, S1 ⇒ S2 is a sentence (implication)

If S1 and S2 are sentences, S1 ⇔ S2 is a sentence (biconditional)

Semantik der Aussagenlogik

In der Aussagenlogik legt ein Modell Wahrheitswerte (true oder false) fest

E.g. m_1 = { $P_{1,2}$ = false, $P_{2,2}$ = true, $P_{3,1}$ = false }

Bei 3 Symbolen gibt es $2^3 = 8$ mögliche Modelle, die automatisch generiert werden können (siehe Wumpus-Beispiel)

Rules for evaluating truth with respect to a model *m*:

¬S	is true iff	S is false	
S1 ∧ S2	is true iff	S1 is true and	S2 is true
S1 ∨ S2	is true iff	S1is true or	S2 is true
S1 ⇒ S2	is true iff	S1 is false or	S2 is true
i.e.,	is false iff	S1 is true and	S2 is false
S1 ⇔ S2	is true iff	S1⇒S2 is true and S2⇒S1 is true	

Simple recursive process evaluates an arbitrary sentence, e.g.,

$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (true \vee false) = true \wedge true = true$

Wahrheitstabelle für log. Verknüpfungen

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Eine einfache Wissensbasis

Wahl des Vokabulars

- Sei $P_{i,j}$ true, wenn sich in (i,j) eine Falltür befindet
- Sei $B_{i,j}$ true, wenn auf (i,j) ein Luftzug vorhanden ist

Die **Wissensbasis** enthält folgende Sätze:

- Es gibt keine Falltür in (1,1)

  R_1: $\neg P_{1,1}$
- Ein Quadrat hat genau dann einen Luftzug, wenn sich auf dem benachbarten Quadrat eine Falltür befindet:

  R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

  R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- Aufnahme der Luftzugwahrnehmungen der ersten 2 Quadrate:

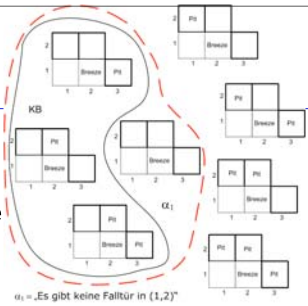
  R_4: $\neg B_{1,1}$

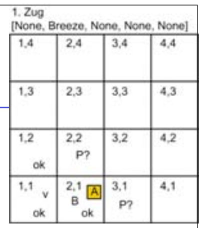
  R_5: $B_{2,1}$

Inferenz

Ziel der logischen Inferenz ist, zu entscheiden, ob KB |= α

Ein Algorithmus für die logische Inferenz könnte alle Modelle auflisten und prüfen, ob α in jedem Modell wahr ist, in dem auch die Wissensbasis wahr ist.





Die relevanten Aussagensymbole in der Wumpuswelt sind hier

$B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,2}$, $P_{3,1}$

Bei 7 Symbolen gibt es 2^7=128 möglich Modelle.

In drei davon ist die Wissensbasis KB wahr. In diesen drei Modellen ist $\neg P_{1,2}$ wahr, weil sich in [1,2] keine Falltür befindet.

$P_{2,2}$ ist in zweien dieser drei Modellen wahr und in einem falsch.

Also können wir nicht erkennen, ob sich in [2,2] ein Falltür befindet.

Die Wissensbasis KB, α_1 und ihre Wahrheitstafel

KB = R_1: $\neg P_{1,1}$

R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

α_1 = „Es gibt keine Falltür in [1,2]"

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	false	false	false	false	false	false	false	true
false	false	false	false	false	false	true	false	true
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	true	true
false	true	false	false	false	true	false	true	true
false	true	false	false	false	true	true	true	true
false	true	false	false	true	false	false	false	true
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	true	true	true	true	true	true	false	false

Bei 7 Symbolden gibt es 2^7=128 mögliche Modelle

Die KB ist nur in drei Modellen wahr! In diesen ist $P_{1,2}$ falsch, d.h. keine Falltür in [1,2].

Der Inferenzalgorithmus ist korrekt, aber unbrauchbar

Da der Inferenz-Algorithmus alle mögliche Modelle aufzählt, müsste er $2^{\text{Anzahl der Symbolde}}$ viele Modelle generieren, also ist seine Zeitkomplexität $O(2^n)$. Der Algorithmus ist also für reale Probleme unbrauchbar.

Jeder bis heute bekannte Inferenz-Algorithmus für die Aussagenlogik hat im schlechtesten Fall eine Komplexität, die exponentiell zur Eingabegröße ist. (also NP-vollständig)

Konzept: Logische Äquivalenz „≡”

Two sentences are logically equivalent iff true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{array}{rcll}
(\alpha \wedge \beta) & \equiv & (\beta \wedge \alpha) & \text{commutativity of } \wedge \\
(\alpha \vee \beta) & \equiv & (\beta \vee \alpha) & \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) & \equiv & (\alpha \wedge (\beta \wedge \gamma)) & \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) & \equiv & (\alpha \vee (\beta \vee \gamma)) & \text{associativity of } \vee \\
\neg(\neg \alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg \beta \Rightarrow \neg \alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg \alpha \vee \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) & \equiv & (\neg \alpha \vee \neg \beta) & \text{de Morgan} \\
\neg(\alpha \vee \beta) & \equiv & (\neg \alpha \wedge \neg \beta) & \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) & \equiv & ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) & \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) & \equiv & ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) & \text{distributivity of } \vee \text{ over } \wedge
\end{array}
$$

Konzept: Gültigkeit (validity)

Ein Satz ist gültig, wenn er in allen Modellen wahr ist.

Beispiel: P $\vee$ ¬P ist gültig

Gültige Sätze werden als Tautologien bezeichnet. Sie sind notwendigerweise wahr und damit semantisch ohne Aussagekraft.

Aus der Definition der logischen Konsequenz kann das Deduktionstheorem abgeleitet werden:

> *Für alle Sätze α und β gilt $\alpha \models \beta$ genau dann, wenn der Satz $(\alpha \Rightarrow \beta)$ gültig ist, d.h. wenn alle Modelle den Satz erfüllen.*

$\Rightarrow$ Implikation (wenn ... dann ...)

$\models$ logische Konsequenz, Gültigkeit

alle Modelle der Welt

Konzept: Erfüllbarkeit (satisfiability)

Ein Satz ist erfüllbar, wenn er in irgendeinem Modell der Welt wahr ist.

Beispiel: die Wissensbasis KB = $(R_1, R_2, R_3, R_4, R_5)$ ist erfüllbar, weil es drei Modelle gibt, in denen sie wahr ist.

Wenn ein Satz α in einem Modell m wahr ist, sagen wir, m erfüllt α oder m ist ein Modell für α.

Die Erfüllbarkeit von Sätzen in der Aussagenlogik, war das erste Problem in der Informatik, das als NP-vollständig nachgewiesen wurde.

Erfüllbarkeitsprobleme in der Informatik

Viele Probleme der Informatik sind Erfüllbarkeitsprobleme. Zum Beispiel sind alle Probleme unter Rand- oder Nebenbedingungen – so genannte Constraint Satisfaction Problems (CSP) – dadurch gekennzeichnet, dass versucht wird, durch irgendeine Zuweisung an Variablen die Randbedingungen zu erfüllen.

Durch geeignete Transformationen können Suchprobleme auch dadurch gelöst werden, indem ihre Erfüllbarkeit geprüft wird.

Gültigkeit vs. Erfüllbarkeit, Widerspruch

Es gilt: Ein Satz α in einem Modell m ist gültig genau dann, wenn $\neg\alpha$ in der Welt unerfüllbar ist.

Es gilt auch der **Umkehrschluss**:

Ein Satz α ist genau dann erfüllbar, wenn $\neg\alpha$ nicht gültig ist.

Praktisches Ergebnis:

$\alpha \models \beta$ genau dann, wenn der Satz $(\alpha \wedge \neg \beta)$ unerfüllbar ist.

wichtig für Resolutions-algorithmus

Beweist man β aus α – d.h. man nimmt an, dass gilt $\beta \rightarrow \alpha$ –, indem man die Unerfüllbarkeit von $(\alpha\wedge\neg\beta)$ überprüft, dann ist das mathematisch gesehen ein Widerspruchsbeweis (reductio ad absurdum)

Inferenzregeln in der Aussagenlogik

Bekannteste Regel: Modus Ponens $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$

Beispiel

$$\frac{(WumpusAhead \land WumpusAlive) \Rightarrow Shoot, WumpusAhead \land WumpusAlive}{Shoot}$$

Weitere Regel: Und-Elimination $\frac{\alpha \land \beta}{\alpha}$

Die Regeln Modus Ponens und die Und-Elimination sind für alle Modelle (Instanzen in einer Welt) korrekt, d.h. sie erzeugen korrekte Inferenzen, ohne dass Modelle aufgelistet werden müssen.

Alle logischen Äquivalenzen können als Inferenzregeln verwendet werden.

Inferenzregeln in der Wumpuswelt anwenden

Betrachte die KB R_1, ..., R_5

KB = R_1: $\neg P_{1,1}$

R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

Beweise: $\neg P_{1,2}$, d.h. in Feld [1,2] befindet sich keine Falltür

Wende die Bikonditional-Eliminierung auf R_2 an. Es folgt:

R_6: $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

Wende die Und-Eliminierung auf R_6 an. Es folgt:

R_7: $((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

Regel R_8 – R_{10}

R_7: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Die logische Äquivalenz für Kontraposition ergibt:

R_8: $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$

Anwendung des Modus Ponens auf R_8 zusammen mit der Wahrnehmung R_4 (d.h. $\neg B_{1,1}$) ergibt:

R_9: $\neg(P_{1,2} \vee P_{2,1}))$

Die Anwendung der de Morganschen Regel ergibt:

R_{10}: $\neg P_{1,2} \wedge \neg P_{2,1}$

weder in $P_{1,2}$ noch in $P_{2,1}$ befindet sich eine Falltür

Suche nach Beweisen vs. Auflistung von Modellen

Die Suche nach Beweisen ist eine Alternative zur Auflistung von Modellen.

Die Suche kann bei der anfänglichen **Wissensbasis** beginnen und von dort aus **vorwärts** erfolgen, wobei die Inferenzregeln angewendet werden, um den **Zielsatz** abzuleiten.

Man kann aber auch beim **Zielsatz** beginnen und von dort aus **rückwärts** gehen, um zu versuchen eine Kette von Inferenzregeln zu finden, die von der anfänglichen **Wissensbasis** zum Zielsatz führt.

Inferenz ist NP-vollständig!!

Effiziente Inferenz?

In bestimmten Fällen kann *die Ermittlung eines Beweise jedoch höchst effizient sein, nämlich dann, wenn irrelevante Aussagensymbole ignoriert werden können, und zwar unabhängig davon, wie viele es davon gibt.*

Der oben gezeigte Beweis, der zu $\neg P_{1,2} \wedge \neg P_{2,1}$ führte erwähnt nicht die Symbole $B_{2,1}$, $P_{1,1}$ $P_{2,2}$ oder $P_{3,1}$. Sie können ignoriert werden, weil die Zielaussage $P_{1,2}$ nur in Satz R_4 erscheint; die anderen Aussagensymbole erscheinen nur in R_4 und R_2. Die Sätze R_1, R_3 und R_5 haben also für den Beweis keine Bedeutung. Alle Aussagensymbole, die in R_1, R_3 und R_5 vorkommen, können somit ignoriert werden.

Monotonie

Die Eigenschaft logischer Systeme folgt aus einer grundlegenderen Eigenschaft, der Monotonie.

Die Monotonie besagt, dass die Menge der folgerbaren Sätze nur dann wachsen kann, wenn der Wissensbasis Informationen hinzugefügt werden.

$$\text{wenn } KB \models \alpha \quad \text{dann} \quad KB \wedge \beta \models \alpha$$

Angenommen, die Wissensbasis enthält die zusätzliche Behauptung β, die aussagt, dass es in der Welt genau 8 Falltüren gibt. Dieses Wissen könnte dem Agenten helfen, zusätzliche Schlüsse zu ziehen, aber er kann keinen Schluss revidieren, den α bereits abgeleitet hat - etwa, dass es in [1,2] keine Falltür gibt.

Monotone Logik problematisch für menschliches Schlussfolgern

Die Übertragung der monotonen Logik auf das menschliche logische Schließen ist problematisch.

Schlussfolgerungen im Alltag habe oft nicht-monotonen Charakter, das bedeutet: Durch später hinzu gefügte Informationen könne frühere Schlussfolgerungen revidiert werden. Zum Beispiel:

Wir wissen	*Tux* ist ein Vogel
	⇒ *Tux* kann fliegen
später erfahren wir:	*Tux* ist ein Pinguin
	⇒ *Tux* kann nicht fliegen

Ein solcher Schluss ist in der monotonen Logik nicht möglich, wohl aber in der nicht-monotonen Logik.

Resolution

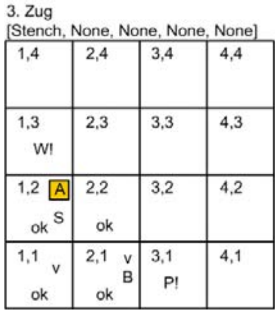
Bisher gezeigt: Korrektheit der Inferenzregeln

Frage: Vollständigkeit von Inferenzalgorithmen

Problem: Wenn die verfügbaren Inferenzregeln ungeeignet sind, ist da ziel nicht erreichbar, d.h. es gibt keinen Beweis (für das Ziel), der nur diese Inferenzregeln verwendet.

Die Inferenzregel *Resolution* ergibt einen vollständigen Inferenzalgorithmus, wenn sie mit einem beliebigen vollständigen Suchalgorithmus kombiniert wird.

Resolutionsregel in der Wumpuswelt



Betrachte die Schritte, die zur folgenden Situation geführt haben: der Agent geht von [2,1] nach [1,1] zurück und dann weiter nach [1,2], wo er einen Gestank S, aber keinen Luftzug B wahrnimmt. Wir fügen dies der Wissensbasis hinzu:

R_{11}: $\neg B_{1,2}$

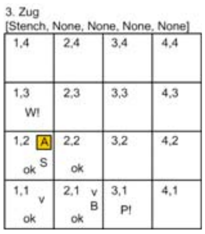
R_{12}: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

Wie der Beweis von R_{10}, beweisen wir jetzt das Fehlen von Falltüren in [2,2] und [1,2]:

R_{13}: $\neg P_{2,2}$

R_{14}: $\neg P_{1,3}$

Resolution in der Wumpuswelt ..



Bikonditional-Eliminierung für R_3, gefolgt von Modus Ponens für R_5 liefert die Tatsache, dass es in [1,1], [2,2] oder [3,1] eine Falltür gibt:

R_{15}: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

Erstmalige Anwendung der Resolutionsregel:

Das Literal $\neg P_{2,2}$ in R_{13} wird mit dem Literal $P_{2,2}$ in R_{15} resolviert:

R_{16}: $P_{1,1} \vee P_{3,1}$

R_{17}: $P_{3,1}$

resolvieren von $\neg P_{1,1}$ in R_1 mit dem Literal $P_{1,1}$ in R_{16}

Einheitsresolution

Die Inferenzregel der Einheitsresolution lautet:

$$\frac{l_1 \vee \ldots \vee l_k, m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

wobei jedes $\boldsymbol{l}$ ein Literal ist und $\boldsymbol{l_i}$ und $\boldsymbol{m}$ komplementäre Literale sind.

Die Einheitsresolutionsregel verwendet eine Klausel – eine Disjunktion von Literalen – und ein Literal und erzeugt daraus eine neue Klausel.

Ein einzelnes Literal heißt auch Einheitsklausel.

Vollständige Resolutionsregel

Die Verallgemeinerung der Einheitsresolution führt zur allgemeinen Resolutionsregel:

Das Literal m_j wird mit dem Literal l_i resolviert

$$\frac{l_1 \vee \ldots \vee l_k, m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

wobei jedes $\boldsymbol{l_i}$ und m_j komplementäre Literale sind.

Klauseln der Länge 2 schreibt man als $\dfrac{l_1 \vee l_2, \neg l_2 \vee l_3}{l_1 \vee l_3}$

Beispiel: $\dfrac{P_{1,1} \vee P_{3,1}, \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}$

Vollständige Inferenz mit Resolution möglich

Die Resolution ist die Grundlage für vollständige Inferenzalgorithmen.

Jeder vollständige Suchalgorithmus, der die Resolutionsregel anwendet, kann jede Schlussfolgerung ableiten, die Konsequenz einer beliebigen aussagenlogischen Wissensbasis ist.

Die Resolution ist Widerlegungsvollständig, d.h. die Resolution kann immer verwendet werden, um einen Satz zu bestätigen oder zu widerlegen, aber sie kann nicht verwendet werden, um wahre Sätze aufzulisten.

Konjunktive Normalform

Jeder Satz der Aussagenlogik ist logisch äquivalent mit einer Konjunktion von Disjunktionen von Literalen.

Ein Satz, der als Konjunktion von Disjunktionen von Literalen ausgedrückt wird, befindet sind in der konjunktiven Normalform (KNF)

Ein k-KNF-Satz hat genau k Literale pro Klausel:

$$(l_{1,1} \vee \ldots \vee l_{1,k}) \wedge \ldots \wedge (l_{n,1} \vee \ldots \vee l_{n,k})$$

Beispiel: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \rightarrow$ KNF

1. Wir eliminieren $\Leftrightarrow$, indem wir $\alpha \Leftrightarrow \beta$ durch $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ ersetzen:

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Wir eliminieren $\Rightarrow$, indem wir $\alpha \Rightarrow \beta$ durch $(\neg\alpha \vee \beta)$ ersetzen:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Bei KNF darf $\neg$ nur in Literalen auftreten: wir verschieben $\neg$ nach innen, indem wir wiederholt die Äquivalenzen „Elimination der doppelten Negation" und „de Morgan" anwenden:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Distributivgesetz löst Disjunktionen von Konjunktionen auf:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolutionsalgorithmus und Widerspruch

Inferenzalgorithmen für die Resolutionsregel arbeiten mit der Beweistechnik des Widerspruchs:

Wir zeigen, dass ein Satz α aus der Wissensbasis KB gefolgert werden kann, d.h. KB $|= \alpha$, indem wir zeigen, dass es kein Modell in KB gibt, welches $\neg\alpha$ erfüllt, d.h. (KB $|= \neg\alpha$) nicht erfüllbar ist. Wir verwenden einen Widerspruchsbeweis.

Ein Resolutionsalgorithmus in der Aussagenlogik

1. (KB $\wedge$ $\neg\alpha$) wird in KNF transformiert
2. Resolutionsregel wird auf die resultierenden Klauseln angewendet. Jedes Paar, das komplementäre Literale enthält, wird aufgelöst (resolviert), um eine neue Klausel zu erzeugen.
3. Die neue Klausel wird zur Klauselmenge hinzugefügt, falls sie noch nicht darin enthalten ist.

Dieser Prozess wird fortgesetzt, bis eines von zwei Dingen passiert:

- Es gibt keine neuen Klauseln, die hinzugefügt werden können, dann ist β nicht aus α folgerbar, oder
- eine Anwendung der Resolutionsregel leitet die *leere Klausel* ab, dann ist β aus α folgerbar.

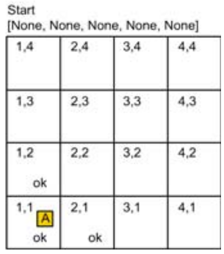
Was bedeutet die leere Klausel?

Leere Klausel

Die leere Klausel - eine Disjunktion ohne Disjunkte - ist äquivalent mit *false*, weil eine Disjunktion nur dann wahr ist, wenn mindestens eines ihrer Disjunkte wahr ist.

Eine andere Möglichkeit zu erkennen, dass eine leere Klausel einen Widerspruch darstellt, ist die Beobachtung, dass sie nur aus der Resolution von zwei komplementären Einheitsklauseln wie etwa P und $\neg$P entsteht.

Resolutionsprozedur in der Wumpus-Welt



Wenn sich der Agent in Feld [1,1] befindet, spürt er keinen Luftzug; es kann also keine Falltür auf benachbarten Quadraten sein.

Die relevante Wissensbasis lautet:

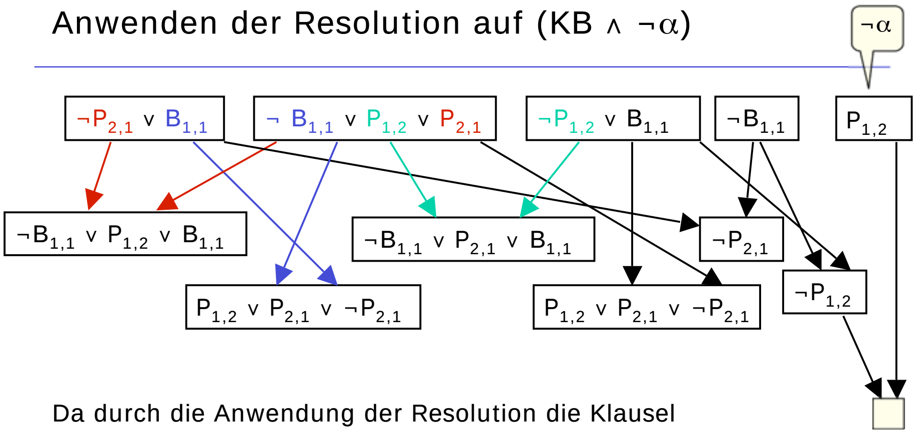
KB = $R_2 \wedge R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

Wir wollen beweisen, dass z.B. $\alpha = \neg P_{1,2}$ gilt.

Zuerst wird (KB $\wedge \neg\alpha$) – das ist dasselbe wie (KB $|= \alpha$) – in KNF umgewandelt:

$(\neg P_{1,2} \vee B_{1,1}) \wedge (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg B_{1,1}) \wedge (P_{1,2})$

Anwenden der Resolution auf (KB $\wedge \neg\alpha$)



Da durch die Anwendung der Resolution die Klausel abgeleitet wird, ist gezeigt, dass (KB $\wedge \neg\alpha$) nicht erfüllbar ist. Die Annahme von $\neg\alpha$ führt zu einem Widerspruch, was besagt, dass $\alpha = \neg P_{1,2}$ wahr ist. Es gibt also in Feld [1,2] keine Falltür.

Resolutionstheorem

Das Vollständigkeitstheorem für die Resolution wird als Resolutionstheorem bezeichnet:

- Wenn eine Menge von Klauseln nicht erfüllbar ist, enthält der Resolutionsschluss dieser Klauseln die leere Klausel.

Das Resolutionstheorem wird dadurch bewiesen, indem das Gegenteil gezeigt wird: Wenn der Abschluss RC(S) die leere Klausel nicht enthält, ist S erfüllbar.

Vorwärts- und Rückwärtsverkettung

Die Vollständigkeit der Resolution macht sie zu einer wichtigen Inferenz-Methode. In vielen praktischen Situationen braucht man nicht die ganze Leistungsfähigkeit der Resolution. Wissensbasen aus der realen Welt enthalten oft nur eingeschränkte Klauseln, so genannte **Horn-Klauseln**.

Eine Horn-Klausel ist eine Disjunktion von Literalen, von denen *höchstens eine positiv* ist.

Beispiel: Die Klausel ($\neg L_{1,1} \lor \neg$ Breeze $\lor B_{1,1}$), wobei $L_{1,1}$ die Position des Agenten auf dem Quadrat [1,1] angibt, ist eine Horn-Klausel, während ($\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}$) das nicht ist.

Die Beschränkung auf Horn-Klauseln hat drei wichtige Gründe!

Horn-Klauseln

Drei wichtige Gründe für Horn-Klauseln:

1. Jede Horn-Klausel kann als Implikation geschrieben werden, deren Prämisse eine Konjunktion positiver Literale und deren Schluss ein einziges positives Literal ist.

   Beispiel: Die Horn-Klausel

   $$(\neg L_{1,1} \vee \neg \text{Breeze} \vee B_{1,1})$$

   kann als Implikation

   $$(L_{1,1} \wedge \text{Breeze}) \Rightarrow B_{1,1}$$

   geschrieben werden.

   Breeze ist die aktuelle Wahrnehmung. $B_{1,1}$ ist die Tatsache (Fakt), dass auf Quadrat [1,1] ein Luftzug besteht.

   Als Implikation ist der Satz einfacher zu lesen: er besagt, dass, wenn ein Agent sich auf Quadrat [1,1] befindet und dort ein Luftzug zu spüren ist, dann ist [1,1] windig.